Qualcomm Technologies, Inc.

Qualcomm Cloud AI 100 Getting Started (SDK 1.9 Release)

User Guide

HD-PT790-991A Rev. 3

February 28, 2023

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:50:29 GMT mixd2@lenovo.com

For additional information or to submit technical questions, go to https://www.qualcomm.com/support

Confidential – Qualcomm Technologies, Inc. and/or its affiliated companies – May Contain Trade Secrets

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to DocCtrlAgent@qualcomm.com.

Confidential Distribution: Use or distribution of this item, in whole or in part, is prohibited except as expressly permitted by written agreement(s) and/or terms with Qualcomm Incorporated and/or its subsidiaries.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc.

All Qualcomm products mentioned herein are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

© 2023 Qualcomm Technologies, Inc. and/or its subsidiaries. All rights reserved.

Contents

I 5
1 Cloud AI 100 introduction 6
1.1 Inference Workflow and Software Stack 6
1.2 Cloud AI 100 products 8
1.3 AIC100 SoC architecture 9
1.4 Platform hardware architecture 10
1.5 Platform software architecture 11
2 Installation Guide 15
2.1 Supported Operating Systems and Platforms 15
2.1.1 Docker 16
2.1.2 KVM 16
2.1.3 Hyper-V 17
2.2 SDK installation guide 19
2.2.1 Prerequisites for Ubuntu 20.04 / 22.04 on x86 19
2.2.2 Prerequisites for Ubuntu 18.04 on x86 20
2.2.3 Prerequisites for Centos 7 / RHEL on x86 21
2.2.4 Platform SDK Installation 22
2.2.5 Apps SDK Installation 24
2.3 Docker and Kubernetes support 27
2.3.1 Qualcomm deliverables 27
2.3.2 QAic docker 28
2.3.3 Kubernetes 30
2.4 Deployment examples 31
2.4.1 Deploy with mini-kube 31
3 Release Notes 34
4 Neural Network Support 35
5 Quick Start Guide 38
5.1 Performance tuning overview 38
5.2 Inference workflow overview 39
5.3 Jupyter notebooks 40

Qualcomm Confidential – May Contain Trade Secrets 2023-03-03 04:50:29 GMT rixu@lenovo.com

Tables

Table 2-1: Supported operating systems and platforms..... 15

Table 2-2: QAic K8s device plugin build script argument details..... 31

Qualcomm
Confidential - May Contain Trade Secrets
2023-03-03 04:50:29 GMT
mixd2@lenovo.com

Figures

Figure 1-1: Cloud AI 100 PCIe HHHL Card 8
Figure 1-2: Cloud AI 100 Dual M.2 Card (with and without heat spreader) 9
Figure 1-3: AIC100 SoC block diagram 10
Figure 2-1: Cloud AI 100 website 22
Figure 2-2: AIC Platform SDK 22
Figure 2-3: Cloud AI 100 website 25
Figure 2-4: Kubernetes cluster 27

Qualcomm Confidential - May Contain Trade Secrets
2023-03-03 04:50:29 GMT
mixd2@lenovo.com

Qualcomm
Confidential - May Contain Trade Secrets
2023-03-03 04:50:29 GMT
mixd2@lenovo.com

1 Cloud AI 100 introduction

Qualcomm Cloud AI 100 accelerator cards and accompanying SDKs offer superior power and performance capabilities to meet the growing inference needs of Cloud Data Centers, Edge, and other machine learning (ML) applications. The Cloud AI 100 card is powered by the AIC100 system-on-chip (SoC), which is specialized for ML inference workloads. The Apps and Platform SDKs provide the ability to compile, optimize, and run deep learning models from popular frameworks such as PyTorch, TensorFlow, ONNX, Caffe, and Caffe2 on Cloud AI 100 hardware.

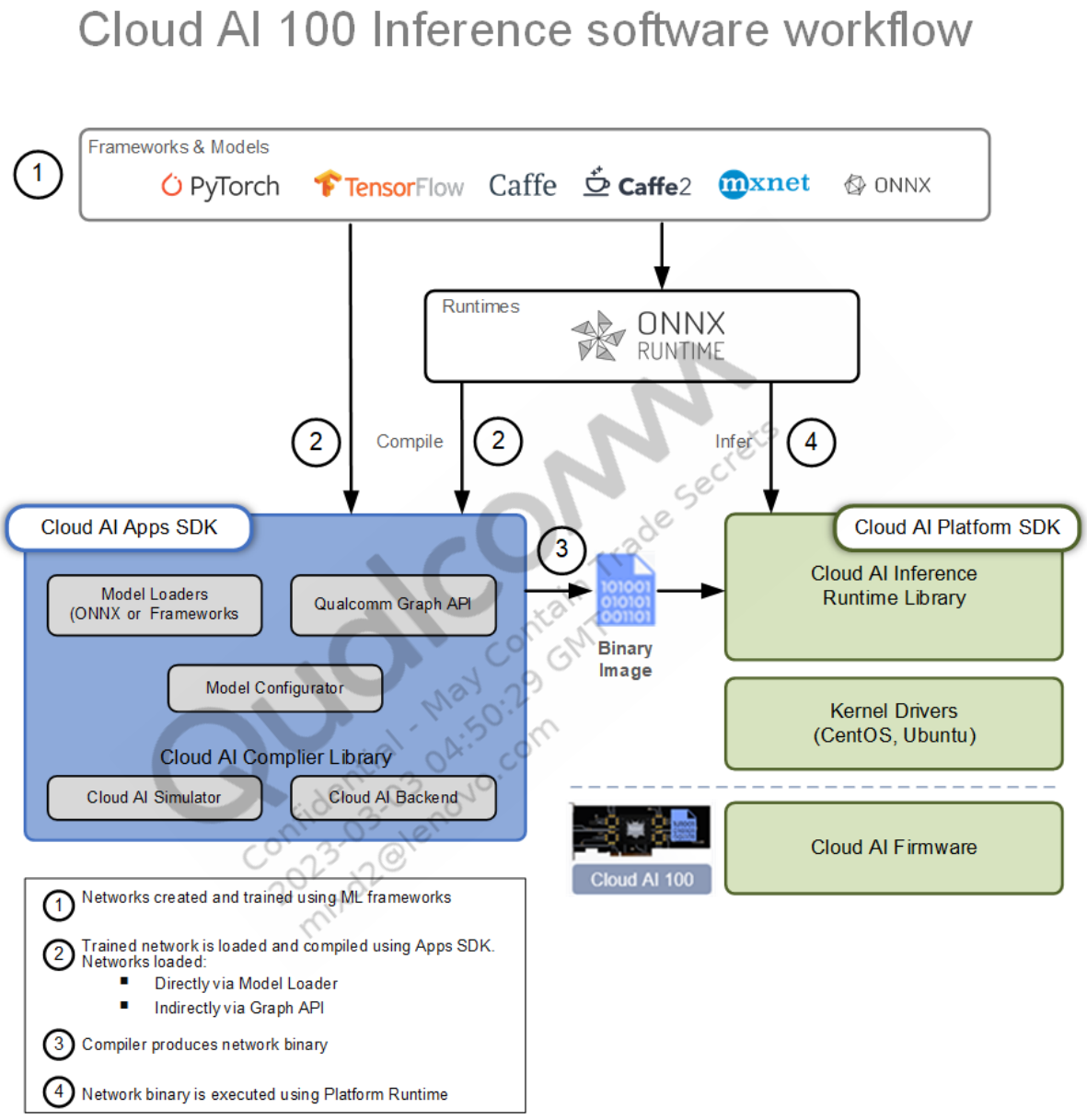
1.1 Inference Workflow and Software Stack

The Cloud AI 100 inference workflow has the following steps at a high level:

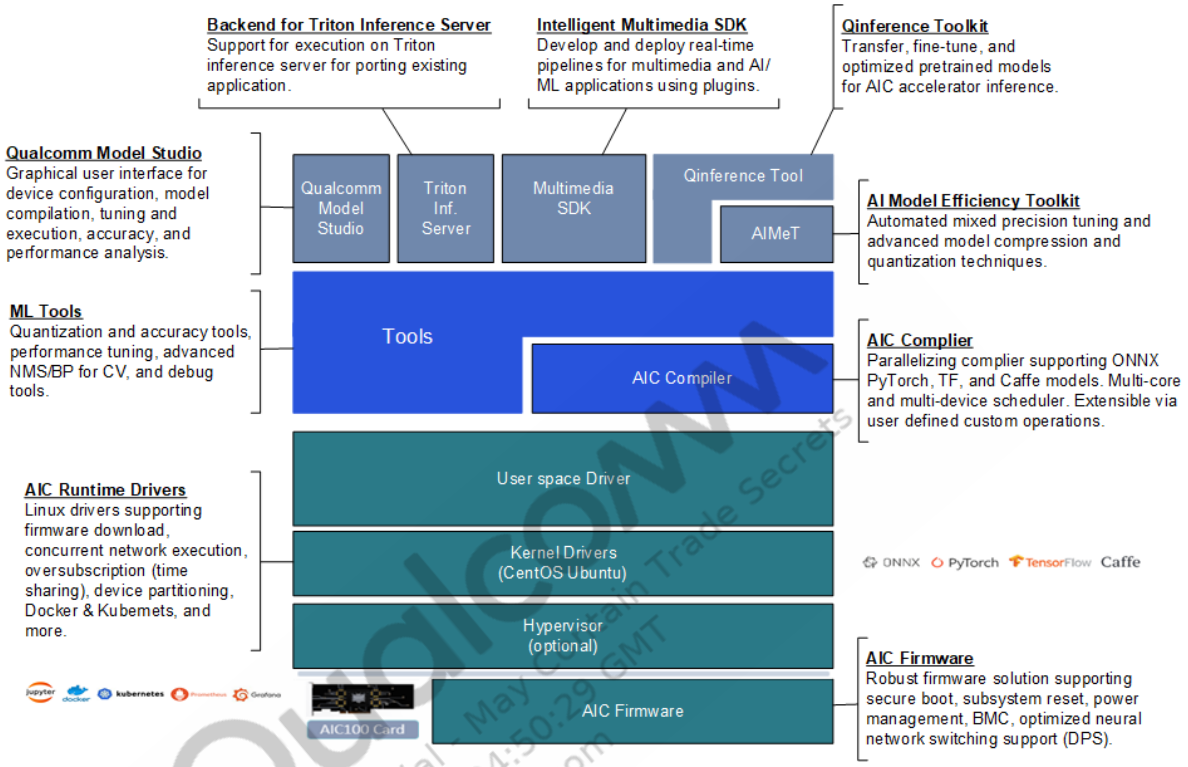
1. An inference-friendly network is generated from the trained neural network created using common ML frameworks.
2. The inference-friendly network is loaded and compiled using the Apps SDK. The network is loaded:
   a. Directly via model loader
   b. Or indirectly via graph API
3. The compiler produces the network binary.
4. The network binary is executed on the Cloud AI 100 hardware using Platform Runtime.

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:50:29 GMT nuvd2@lenovo.com

The following figure shows the Cloud AI 100 inference workflow.



The Cloud AI 100 software stack provides a comprehensive suite of developer tools for ML application developers and system integrators to create and deploy ML inference solutions across various use cases. The following figure highlights the comprehensive full stack solution.


Backend for Triton Inference Server
Support for execution on Triton inference server for porting existing application.
Intelligent Multimedia SDK
Develop and deploy real-time pipelines for multimedia and AI/ML applications using plugins.
Qinference Toolkit
Transfer, fine-tune, and optimized pretrained models for AIC accelerator inference.
Qualcomm Model Studio
Graphical user interface for device configuration, model compilation, tuning and execution, accuracy, and performance analysis.
Qualcomm Model Studio
Triton Inf. Server
Multimedia SDK
Qinference Tool
AIMeT
AI Model Efficiency Toolkit
Automated mixed precision tuning and advanced model compression and quantization techniques.
ML Tools
Quantization and accuracy tools, performance tuning, advanced NMS/BP for CV, and debug tools.
Tools
AIC Compiler
AIC Complier
Parallelizing complier supporting ONNX PyTorch, TF, and Caffe models. Multi-core and multi-device scheduler. Extensible via user defined custom operations.
AIC Runtime Drivers
Linux drivers supporting firmware download, concurrent network execution, oversubscription (time sharing), device partitioning, Docker & Kubernets, and more.
User space Driver
Kernel Drivers (CentOS Ubuntu)
Hypervisor (optional)
ONNX PyTorch TensorFlow Caffe
AIC100 Card
AIC Firmware
AIC Firmware
Robust firmware solution supporting secure boot, subsystem reset, power management, BMC, optimized neural network switching support (DPS).

1.2 Cloud AI 100 products

Qualcomm Cloud AI 100 accelerator cards are available in two form factors – Dual M.2 and PCIe HHHL. The product brief is at https://www.Qualcomm.com/products/technology/processors/cloud-artificial-intelligence/cloud-ai-100. Multiple card SKUs are available in each form factor.

Figure 1-1 Cloud AI 100 PCIe HHHL Card

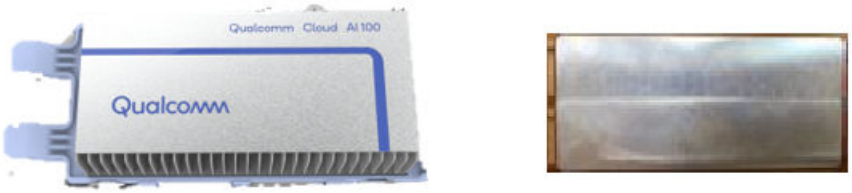


Figure 1-2 Cloud AI 100 Dual M.2 Card (with and without heat spreader)

See the following documents for information regarding the pinout, electrical, mechanical, and thermal specifications:

- *Qualcomm Cloud AI 100 Edge Dual M.2 Card Specification and User Guide* (80-PM720-50)
- *Qualcomm Cloud AI 100 PCIe Half-Height Half-Length Card Specification and User Guide* (80-PM720-48)

1.3 AIC100 SoC architecture

The AIC100 SoC is a high-performance, low-power system for deep learning inference in cloud and edge. It is a highly scalable and power-efficient SoC that can be used across a wide range of thermal design power (TDP) applications. The AIC100 SoC has the following key features:

- Homogenous multicore architecture with up to 16 identical AI cores, also referred to as Neural Signal Processing (NSP) cores
  - An NSP is the smallest computational unit from an application developer point of view.
  - Each AI core consists of a scalar processing unit, four vector processing units, a dual matrix processing unit, and a vector tightly coupled on-chip memory (VTCM) that can be accessed by all three processing units:
    - Scalar processing unit – Multithreaded scalar core with a rich instruction set; supports FP32, FP16, Int16, and Int8 precision
    - Vector processing unit – Rich instruction set for AI, CV, and image processing; supports FP32, FP16, Int16, and Int8 precision
    - Matrix processing unit – High-performance and low-power accelerator for linear algebra (MAC units); supports FP16 and Int8 precision
- One management controller (QSM)
- Peak TOPS
  - 400+ Int8, 200+ FP16
- Up to 144 MB of on-chip memory
  - Reduces DDR memory bandwidth and power
  - Enables on-chip storage of entire weights for many networks

- High-speed Network on Chip (with multicast support)
  - Enables splitting deep learning network operators across AI cores
  - Deep learning network activations are shared via multicast
- 8-lane PCIe Gen4 to interface to host CPU
- LPDDR4x memory with bandwidth up to 136 GB/s
- Secure boot
- Reliability – ECC
- Power management – transient, peak, thermal

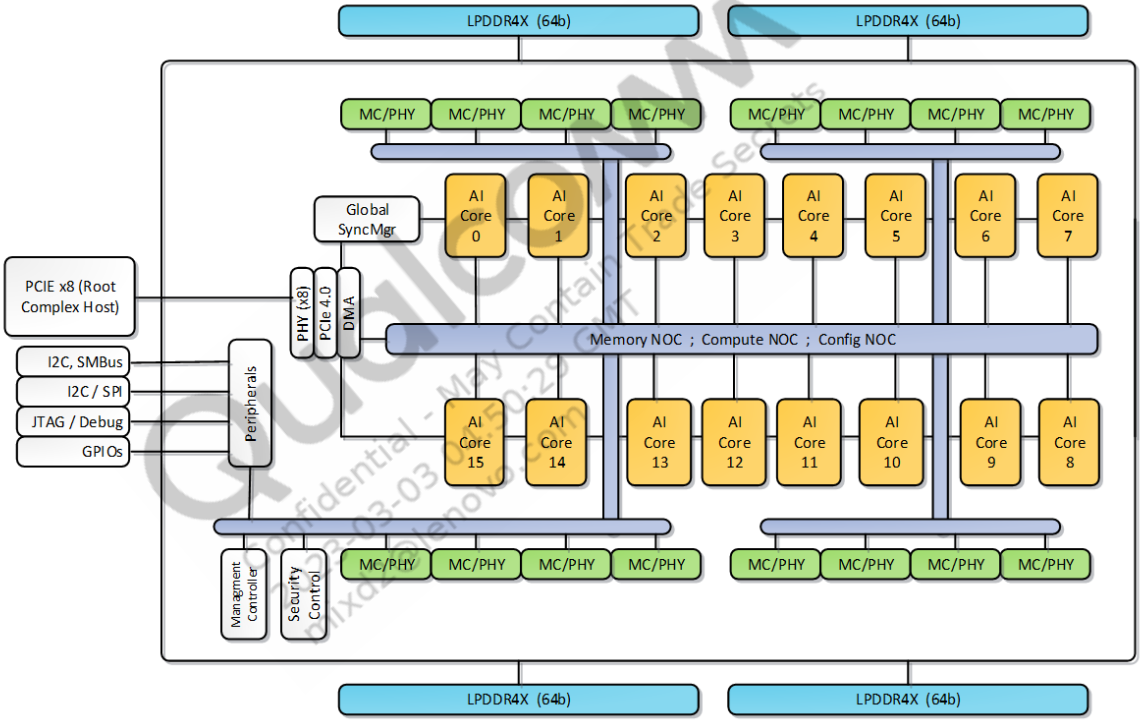


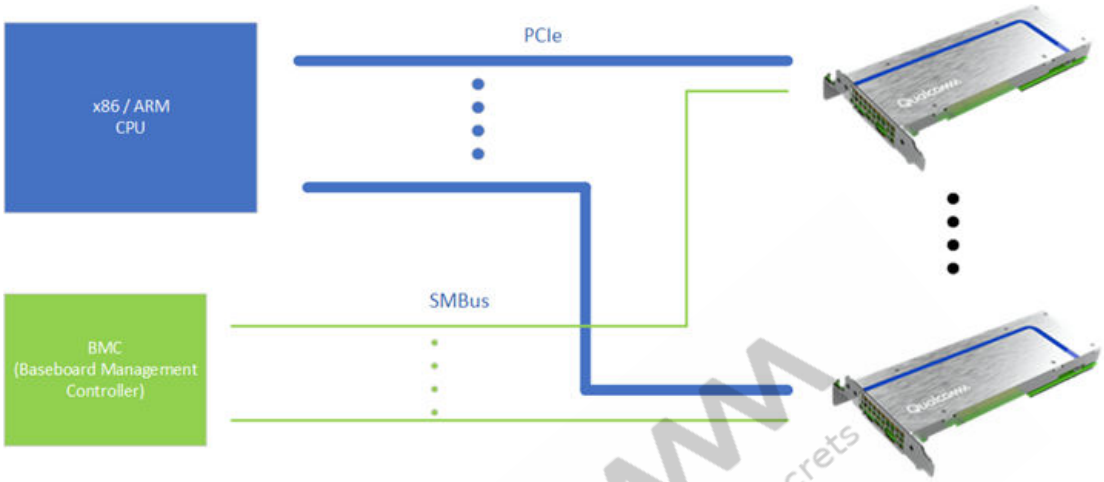
Figure 1-3 AIC100 SoC block diagram

1.4 Platform hardware architecture

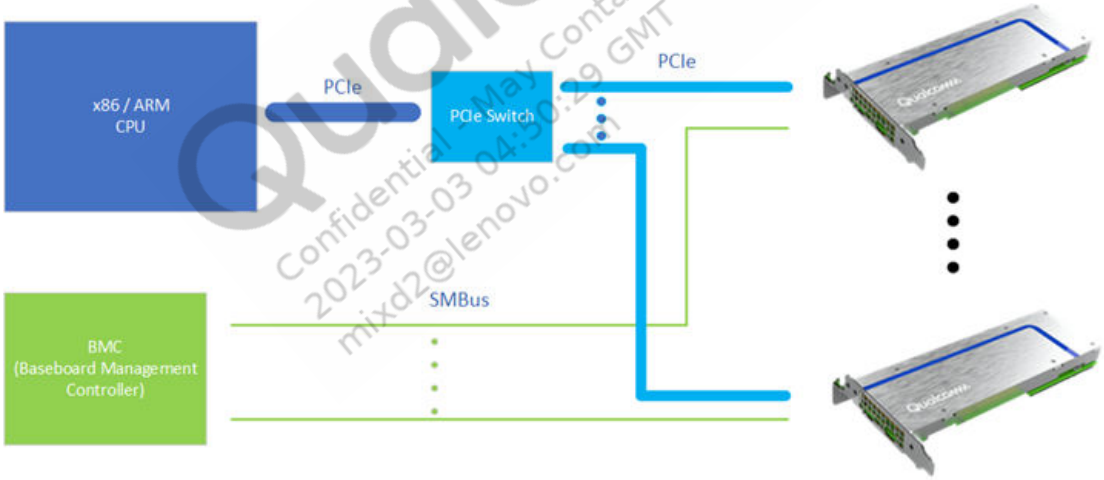
The platform consists of an Arm/x86 CPU connected to one or more Cloud AI 100 accelerator cards over the PCIe bus with or without a PCIe switch in between. A Linux-based host CPU runs applications, runtime libraries, and a kernel driver to communicate with and run inference workloads on the Cloud AI 100 AI accelerator cards.

An SMBus port per Cloud AI 100 card is available for baseboard management controller (BMC) communication.

The following figure shows the platform hardware diagram without a PCIe switch:



The following figure shows the platform hardware diagram with a PCIe switch:



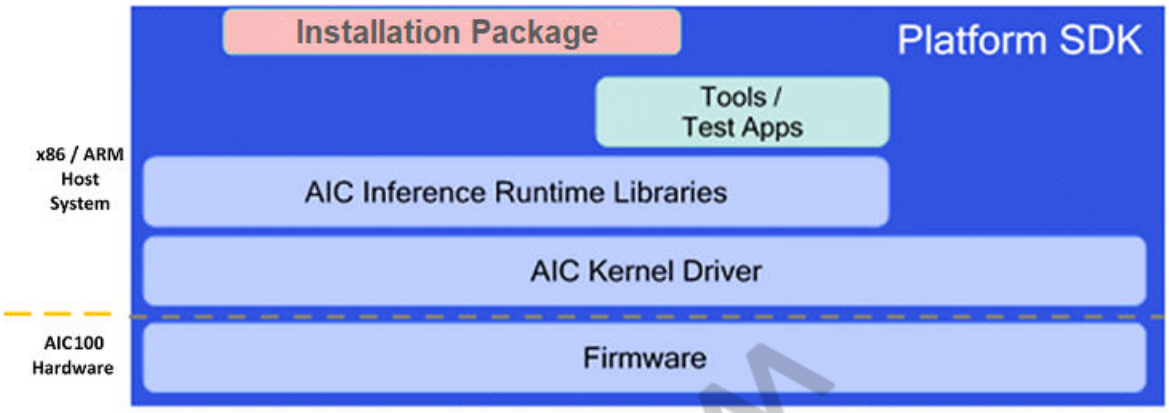
Qualcomm Confidential – May Contain Trade Secrets 2023-03-03 04:30:29 GMT mixd2@lenovo.com

1.5 Platform software architecture

The Platform SDK for Cloud AI 100 (AIC100) includes firmware images for AIC100 hardware, installation scripts, a kernel driver, AIC inference runtime libraries, and test apps and tools for 64-bit x86 and 64-bit Arm-based host systems. The Platform SDK provides functionality for running compiled neural networks, test apps for validating software installation, and hardware functionality and tools for managing, monitoring, profiling, and capturing card and log information.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

See *Getting Started* > *Installation Guide* for information on supported OS/Platforms, installation, and deployment guide.



Cloud AI 100 firmware

Several subsystems (QSM, NSP, etc.) on the AIC100 SoC require firmware images that are loaded during the SoC boot. The key functions of the AIC100 firmware are as follows:

- Secure boot
- Thermal and power control
- Card memory and I/O management
- Inference state management (network load, activate, de-activate)
- Diagnostics and monitoring of SoC resources
- Baseband management controller communication

Based on the card SKU, AI 100 boot could be flash-less or hybrid. Here is a high-level description of the boot flow.

In both cases, boot starts with the execution of the Primary Boot Loader (PBL), which resides within the SoC when the card is provided power. The main difference between flash-less and hybrid boot is as follows:

- Flash-less boot: PBL downloads the Secondary Boot Loader (SBL) from the host over PCIe. At this time, the card is unable to respond to out-of-band (OOB) requests for telemetry.
- Hybrid boot: PBL downloads the SBL from the on-board Flash. At this time, minimum telemetry is available – the card is able to respond to OOB requests for certain telemetry.

In both boot cases, after SBL firmware completes execution, the remaining firmware images – AI 100 Service Manager Runtime Firmware (QSM RT FW) and Neural Signal Processor Root Image Firmware (NSP RI FW) – are downloaded from the host over PCIe and executed on AI 100. This completes the card boot in both cases.

Cloud AI 100 security

Cloud AI 100 device hardware and software security provides protection against internal and external attacks whether it is deployed in a data center or at the edge. Typical attacks would be:

- Booting with unauthorized firmware
- Installation of persistent malware
- Malicious or misbehaving inference jobs corrupting system firmware or other inference jobs
- Malicious or misbehaving host applications corrupting inference jobs owned by other host applications.
- Unauthorized use of debug features
- Devices running with obsolete firmware
- Use of counterfeit AIC100 SoCs

Key hardware security features include:

- Hardware root of trust to achieve secure boot
  - AIC100 SoC ROM is the immutable root of trust
  - Robust extension of root of trust cryptographically
- Hardware access control mechanisms
  - Memory protection units for PCIe DMA transactions
  - Address translation unit (ATU) constrained 3 BARs for memory mapped I/O
  - Compute and memory management unit to sandbox neural net applications

Key software security features include:

- Flash-less and hybrid secure boot
- Rollback protection for firmware
- Ability to enable debug on a secure boot enabled device using a debug policy

Cloud AI 100 Linux runtime

Cloud AI 100 Linux runtime is the software stack that runs on the Linux host. It provides the software interface for inference applications and management/monitoring tools for Cloud AI 100 devices.

The Linux runtime is composed of two major components:

- User space library

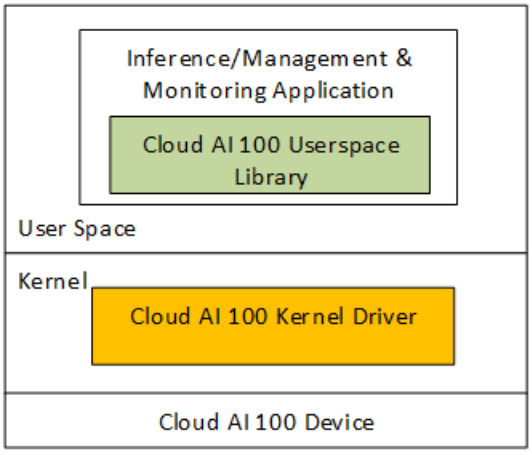
  The user space library is a shared library loaded by the inference application and management/monitoring applications. Applications use the APIs exposed in this library to communicate with the Cloud AI 100 device through the kernel driver.

- Kernel driver

  The Cloud AI 100 kernel driver is a software component that manages Cloud AI 100 hardware in the Linux kernel and exposes interfaces to user space applications through the user space library.

Qualcomm Confidential – May Contain Trade Secrets 2023-03-03 04:50:20 GMT mixed@lenovo.com

The following simplified block diagram shows the two components in the context of the overall software environment:



Qmonitor for device management

Qmonitor is a client-server tool that provides infrastructure to configure and get status information from Cloud AI 100 devices. The main interface to Qmonitor is protocol buffers. It is built with Protocol Buffer 3, allowing for bindings to be generated for the client in any of the supported languages – C++ and Python.

Qmonitor supports encapsulated services for direct messages (command-response) or asynchronous services (register and receive async callbacks).

Qmonitor can be used for

- Device information/configuration (asset tracking, resource availability/utilization, thermal, etc.)
- Device control (reset)
- RAS (memory, I/O)
- Logging
- Sysmon for performance monitoring

Qualcomm Confidential May Contain Trade Secrets 2023-03-03 04:59:59 GMT mixd2@lenovo.com

2 Installation Guide

The Platform (x86 and Arm) and Apps SDK (x86 only) are targeted for Linux-based platforms. The SDKs can be installed natively on Linux operating systems. Container and orchestration are also supported through Docker and Kubernetes. Virtual machines, including KVM and Hyper-V, are also supported. This section covers:

- Installation of the SDKs across three Linux distributions (Centos, Ubuntu, Red Hat)
- Building a docker image with the SDKs and third-party packages for a seamless execution of QAic inference tools/workflow
- Setting up KVM and Hyper-V and installation of SDK
- Deployment with Kubernetes

2.1 Supported Operating Systems and Platforms

The AIC100 Platform SDK is compatible with the following operating systems (OS) and platforms.

Table 2-1 Supported operating systems and platforms

Operating systems	Kernel	X86	Arm™ (Aarch64)
CentOS Linux 7	Linux Kernel 5.4.1	✔	✗
CentOS Linux 8	Linux Kernel 4.19	✗	✔
Ubuntu 18.04	Linux Kernel 5.4.1 / Default Kernel	✔	✗
Ubuntu 20.04	Default Kernel	✔	✔[1]
Ubuntu 22.04	Default Kernel	✔	✗
Red Hat Enterprise Linux 7.9	Default Kernel	✔	✗
Red Hat Enterprise Linux 8.3	Default Kernel	✔	✗
Red Hat Enterprise Linux 8.4	Default Kernel	✔	✗
Red Hat Enterprise Linux 8.6	Default Kernel	✔	✗
Red Hat Enterprise Linux 8.7	Default Kernel	✔	✗
Red Hat Enterprise Linux 9.0	Default Kernel	✔	✗
Red Hat Enterprise Linux 9.1	Default Kernel	✔	✗

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:50:25 GMT mixd2@lenovo.com

Table 2-1 Supported operating systems and platforms (cont.)

Operating systems	Kernel	X86	Arm™ (Aarch64)
AWS linux2	Default Kernel	✔	✗
[1]Supported on certain Arm-based Qualcomm platforms.			

NOTE
- Arm is a trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
- Apps SDK is available only for x86 platforms.

2.1.1 Docker

Docker is a product that allows users to build, test, and deploy applications through software containers. Docker for Cloud AI 100 packages the Platform SDK, Apps SDK (x86 only), libraries, system tools, etc., which enables the user to navigate the inference workflow seamlessly. The building of the docker image and deploying of the containers is discussed in the *Docker and Kubernetes support* section.

2.1.2 KVM

Kernel-based Virtual Machine (KVM) is a module in the Linux kernel that allows the Linux OS to operate as a hypervisor. This enables a native install of Linux to act as the host for a virtual machine. In combination with QEMU and libVirt, a KVM virtual machine emulates a completely independent system that can be limited to a subset of the host hardware, and also can run completely different operating systems from the host.

An overview of KVM is at https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine.

The benefit of a virtual machine is strong isolation. A virtual machine is isolated from other virtual machines and also from the host. This provides a high level of security because one application in a virtual machine may not be aware that it is in a virtual machine, much less that there may be other virtual machines with other applications. Also, a virtual machine provides separation from the host. Even if a driver in the virtual machine crashes the entire virtual machine, that crash will not take down the host and, therefore, will allow other services in other virtual machines to continue operating.

The cost of these benefits is additional overhead to set up the system. Also, additional processing may be required at runtime between the virtual machine and the native hardware.

AIC100 only supports PCIe passthrough to a virtual machine. This means that the virtual machine completely owns the AIC100 device. The AIC100 device cannot be shared between virtual machines, or between a virtual machine and the native host.

The generic setup and operation of a KVM virtual machine is outside the scope of this document. This document assumes that the reader is familiar with those operations and, thus, only explains the elements directly related to assigning an AIC100 device to a KVM virtual machine.

AIC100 supports only the operating systems listed in Table 2-1 as the operating system running within the virtual machine as the guest OS.

Within a virtual machine, AIC100 still requires the assignment of 32 Message Signal Interrupts (MSI) to operate. This requires that the virtual machine emulate an IOMMU. During the creation of a virtual machine, the virtual machine must be configured to emulate a system that can emulate an IOMMU. One such system is the “q35” system. If using “`virt-install`”, a q35 based virtual machine can be created by adding “`–machine=q35`” to the `virt-install` command.

The remainder of this section assumes that the virtual machine is created with `virt-install` and the `-machine=q35` option. Other systems may require different configurations than what is described to obtain the same end effect.

After the virtual machine is created, it must be configured. This can be done with the "`virsh edit`" command while the virtual machine is not running. See the `virsh` man page for more details on this command: https://linux.die.net/man/1/virsh.

The virtual machine configuration must have the emulated IOMMU presented to the guest OS in the virtual machine, and the virtual machine must have the AIC100 device passed in.

First, to present the IOMMU to the guest OS, add the following elements to the configuration XML:

1. Configure the ioapic driver in split mode for interrupt remapping. This is done by adding "`<ioapic driver='qemu'/>`" under the "features" section of the XML.
2. Configure the emulated IOMMU to be present with the interrupt remapping functionality enabled. This is done by adding the following snippet under the "features" section of the XML:

```
<iommu model='intel'>
   <driver caching_mode='on' intremap='on'/>
</iommu>
```

Second, configure what device to pass through to the virtual machine guest OS.

1. Obtain the PCIe SBDF address of the AIC100 device using "`lspci`" in the host.
2. Add the device address obtained from Step 1 to the "devices" section of the XML, as follows, but change the address tag values to match that of your specific system.

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
     <address domain='0x0' bus='0x0' slot='0x19' function='0x0'/>
  </source>
</hostdev>
```

After making these edits, save the configuration. The next time the virtual machine is booted, you will observe the AIC100 device in lspci output in the virtual machine. It will have a different PCIE SBDF address than the host. Install the AIC100 Platform and Apps SDK in the same manner as if you were operating a native system.

2.1.3 Hyper-V

Hyper-V is a virtualization hypervisor commonly used with Microsoft Windows Server. This enables a native install of Windows Server to act as the host for a virtual machine. The virtual machine emulates a completely independent system that can be limited to a subset of the host hardware, and also can run completely different operating systems from the host.

An overview of Hyper-V is at https://en.wikipedia.org/wiki/Hyper-V.

The benefit of a virtual machine is strong isolation. A virtual machine is isolated from other virtual machines and also from the host. This provides a high level of security because one application in a virtual machine may not be aware that it is in a virtual machine, much less that there may be other virtual machines with other applications. Also, a virtual machine provides separation from the host. Even if a driver in the virtual machine crashes the entire virtual machine, that crash will not take down the host and, therefore, will allow other services in other virtual machines to continue operating.

The cost of these benefits is additional overhead to setup the system. Also, additional processing may be required at runtime between the virtual machine and the native hardware.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

AIC100 only supports PCIe passthrough to a virtual machine. This means that the virtual machine completely owns the AIC100 device. The AIC100 device cannot be shared between virtual machines, or between a virtual machine and the native host.

The generic setup and operation of a Hyper-V virtual machine is outside the scope of this document. This document assumes that the reader is familiar with those operations and, thus, only explains the elements directly related to assigning an AIC100 device to a Hyper-V virtual machine.

AIC100 supports only the operating systems listed in Table 2-1 as the operating system running within the virtual machine as the guest OS.

Within a virtual machine AIC100 still requires the assignment of 32 Message Signal Interrupts (MSI) to operate. Hyper-V does not support emulating an IOMMU in the guest OS of the virtual machine. However, Hyper-V supports a paravirtualized PCIe root controller that has a driver in the Linux kernel. This driver uses the Hyper-V hypervisor to configure the host IOMMU for the purposes of supplying MSIs for devices like AIC100.

AIC100 requires the following fixes to the Hyper-V PCIe root controller driver within the Linux kernel to operate properly:

- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?h=v5.19-rc7&id=08e61e861a0e47e5e1a3fb78406afd6b0cea6b6d
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?h=v5.19-rc7&id=455880dfe292a2bdd3b4ad6a107299fce610e64b
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?h=v5.19-rc7&id=b4b77778ecc5bfbd4e77de1b2fd5c1dd3c655f1f
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?h=v5.19-rc7&id=a2bad844a67b1c7740bda63e87453baf63c3a7f7

Consult the provider of your Linux distribution to confirm that those fixes are present in the Linux kernel used for your Linux distribution.

Once the Hyper-V virtual machine is created, the AIC100 device must be assigned to the virtual machine. Hyper-V calls this Direct Device Assignment (DDA) and it is the same as PCIe passthrough.

Assign the AIC100 device to the Hyper-V virtual machine as follows:

1. Configure the virtual machine stop action.
2. Disable the device in Windows.
3. Unmount the device from Windows.

   NOTE This step requires the “-force” option to be used. AIC100 currently does not have a partitioning driver. This may be provided in the future.

4. Add the device to the virtual machine.

Details on these steps can be found at the following DDA resources:

- https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/deploy/deploying-graphics-devices-using-dda
- https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/deploy/deploying-storage-devices-using-dda

To identify the AIC100 device to disable and dismount, look for a device that has the following in the “Hardware Ids” property:

```
PCI\VEN_17CB&DEV_A100&SUBSYS_A10017CB&REV_00
```

Once the AIC100 device is assigned to the virtual machine, it should appear in lspci output in the virtual machine the next time the virtual machine is booted. It will have a different PCIE SBDF address than the host. Install the AIC100 software in the same manner as if you were operating a native system.

2.2 SDK installation guide

NOTE The QAic Kernel Driver can optionally be rebuilt using the host's existing Kernel tree.

The QAic Kernel Driver requires 32 message signaled interrupts (MSI) to operate. On x86-based host systems, Intel VT-d or IOMMU features must be enabled in the BIOS to enable the required number of MSIs.

- For host systems using Intel chipsets, make sure Intel Virtualization (VT-d) is enabled in the BIOS.
- For host systems using AMD chipsets, make sure the IOMMU feature is enabled in the BIOS.

Based on the operating system, certain prerequisites are required. Refer to the following sections for OS-related prerequisites.

2.2.1 Prerequisites for Ubuntu 20.04 / 22.04 on x86

NOTE Ensure that Python 3.8 has been installed on the system.

- Ubuntu 20.04 / 22.04 with default kernel installed, with minimum options.
- Install the "dkms" package and the "linux-headers" package for the kernel.

```
sudo apt-get install dkms
```

- Install the following packages:

```
sudo apt-get update && sudo apt-get install -y software-properties-common
sudo add-apt-repository ppa:ubuntu-toolchain-r/test -y
sudo apt-get update && sudo apt-get install -y build-essential git vim
libpci-dev libudev-dev python3 python3-pip python3-venv python3-setuptools
python3-wheel python3-dev python3.8 python3.8-dev python3.8-venv
sudo apt-get update && sudo apt-get install -y unzip wget ca-certificates
sudo pciutils libglib2.0-dev libssl-dev snap snapd libgl1-mesa-glx openssh-
server pkg-config clang-format libpng-dev
sudo apt-get install libstdc++6
sudo apt-get install libncurses5
sudo pip3 install --upgrade pip
sudo pip3 install wheel numpy opencv-python onnx
```

- Add/update Environment variables:

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/opt/qti-aic/dev/lib/x86_64"
export PATH="/usr/local/bin:$PATH"
export PATH="$PATH:/opt/qti-aic/tools:/opt/qti-aic/exec:/opt/qti-aic/
scripts"
export QRAN_EXAMPLES="/opt/qti-aic/examples"
export QAIC_APPS="/opt/qti-aic/examples/apps"
export QAIC_LIB="/opt/qti-aic/dev/lib/x86_64/libQAic.so"
export QAIC_COMPILER_LIB="/opt/qti-aic/dev/lib/x86_64/libQAicCompiler.so"
```

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:50:39 GMT nuxcole@lenovo.com

2.2.2 Prerequisites for Ubuntu 18.04 on x86

NOTE Ensure that Python 3.8 has been installed on the system.

Install the "`dkms`" package and the "`linux-headers`" package for the kernel.

```
sudo apt-get install dkms
```

The `linux-headers` package is already installed for the stock kernel, but for the 5.4.1 kernel, the headers package needs to be manually installed.

- Download kernel generic .deb files from https://ci.linaro.org/job/lt-qcom-linux-aic100-bionic/lastSuccessfulBuild/artifact/out/ and install as follows:

```
sudo dpkg -i linux-headers-5.4.1-050401-generic_5.4.1-050401-
generic-1_amd64.deb
sudo dpkg -i linux-image-5.4.1-050401-generic_5.4.1-050401-
generic-1_amd64.deb
```

- Install the following packages:

```
sudo apt-get update && sudo apt-get install -y software-properties-common
sudo add-apt-repository ppa:ubuntu-toolchain-r/test -y
sudo apt-get update && sudo apt-get install -y build-essential git vim
libpci-dev libudev-dev python3 python3-pip python3-venv python3-setuptools
python3-wheel python3-dev python3.8 python3.8-dev python3.8-venv
sudo apt-get update && sudo apt-get install -y unzip wget ca-certificates
sudo pciutils libglib2.0-dev libssl-dev snap snapd libgl1-mesa-glx openssh-
server pkg-config clang-format libpng-dev
sudo apt-get install -y libstdc++6
sudo apt-get install -y libncurses5
sudo pip3 install --upgrade pip
sudo pip3 install wheel numpy opencv-python onnx
```

- Add/update Environment variables:

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/opt/qti-aic/dev/lib/x86_64"
export PATH="/usr/local/bin:$PATH"
export PATH="$PATH:/opt/qti-aic/tools:/opt/qti-aic/exec:/opt/qti-aic/
scripts"
export QRAN_EXAMPLES="/opt/qti-aic/examples"
export QAIC_APPS="/opt/qti-aic/examples/apps"
export QAIC_LIB="/opt/qti-aic/dev/lib/x86_64/libQAic.so"
export QAIC_COMPILER_LIB="/opt/qti-aic/dev/lib/x86_64/libQAicCompiler.so"
```

QUALCOMM Confidential and Proprietary – Qualcomm Technologies, Inc. May Contain Trade Secrets

2.2.3 Prerequisites for Centos 7 / RHEL on x86

Use 'sudo' prefix as required.

- Centos 7 / RHEL with default kernel installed
- Run all commands below as ‘root’.
- Install the following packages:
  - `yum update -y`
  - [Optional] `update-ca-trust force-enable`
  - `yum -y install gcc-toolset-9-gcc gcc-toolset-9-gcc-c++ gcc-toolset-9-make gcc-toolset-9-binutils automake`
  - `source /opt/rh/gcc-toolset-9/enable`
  - `yum install -y git vim cmake pciutils-devel rpm-build systemd-devel libudev-devel dkms python3 python3-pip python3-setuptools python3-wheel python3-devel`
  - install dkms
    - Centos 7 – `yum -y install dkms`
    - RHEL – Refer to https://docs.fedoraproject.org/en-US/epel/#How_can_I_use_these_extra_packages.3F on how to install EPEL packages
  - `yum install -y unzip wget ca-certificates sudo pciutils libnsl ncurses-compat-libs libasan glib2-devel ncurses-compat-libs mesa-libGL libpng-devel`
  - `pip3 install --upgrade pip`
  - `pip3 install numpy opencv-python onnx wheel`

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

2.2.4 Platform SDK Installation

- Log in to https://www.qualcomm.com/products/technology/processors/cloud-artificial-intelligence/cloud-ai-100#Overview. First time users need to register.

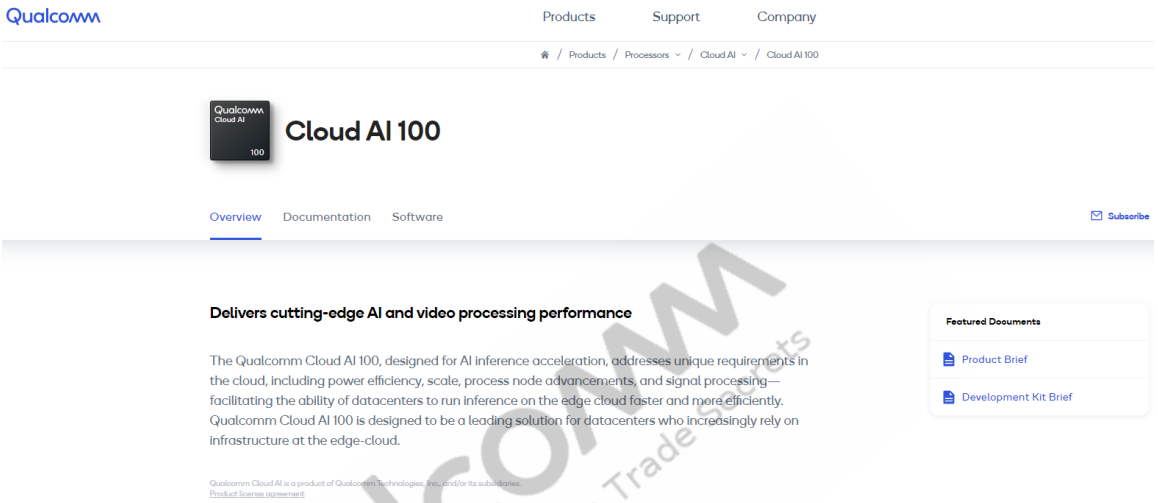

Qualcomm
Products
Support
Company
Products / Processors / Cloud AI / Cloud AI 100
Cloud AI 100
Overview
Documentation
Software
Subscribe
Delivers cutting-edge AI and video processing performance
The Qualcomm Cloud AI 100, designed for AI inference acceleration, addresses unique requirements in the cloud, including power efficiency, scale, process node advancements, and signal processing—facilitating the ability of datacenters to run inference on the edge cloud faster and more efficiently. Qualcomm Cloud AI 100 is designed to be a leading solution for datacenters who increasingly rely on infrastructure at the edge-cloud.
Featured Documents
Product Brief
Development Kit Brief

Figure 2-1 Cloud AI 100 website

- Click on the Software tab.
- Under software tools, click on ‘Cloud AI (AIC) Platform Software Development Kit (SDK)’. Then select the x86/ARM/Ubuntu/Centos Platform SDK and download.

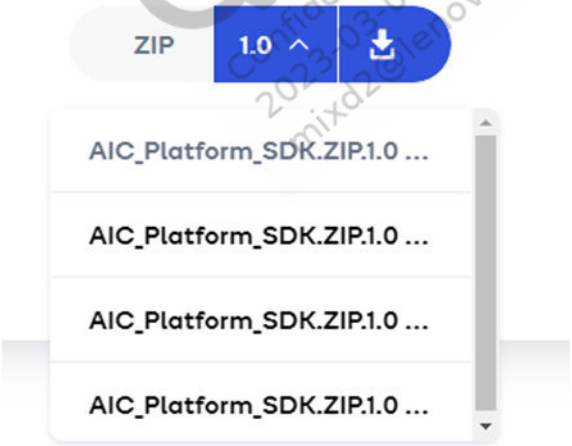


Figure 2-2 AIC Platform SDK

NOTE ‘Software packages’ may provide additional versions of Platform SDK that can be downloaded and installed.

You must be able to log in as root or use `sudo` to have root permissions to complete installation.

NOTE For installation of a new version of the Platform SDK, the previous version must first be uninstalled, followed by sync:

        - `sudo ./uninstall.sh`
        - `sync`

- Copy the Platform SDK downloaded from the Qualcomm Portal to the host machine:
    - For networked x86 or Arm host:
        - Use scp, rsync, or samba to copy the Platform SDK zip file to the host machine
        - Log-in to the host machine (ssh or local terminal)
        - Unzip the downloaded zip file to a working directory
        - `cd` to the working directory
    - For ARM hosts that support Google Android Debug Bridge (ADB):
        - `adb push <Platform SDK zip file> /data`
        - `adb shell`
        - `cd /data`
        - Unzip the downloaded zip file to a working directory
        - `cd` to the working directory

        NOTE When extracting the downloaded zip file, it may contain another embedded zip file. Keep extracting until the "`x86_x64`" or "`aarch64`" folder structure appears.

- Install the Platform SDK:

  The Platform SDK is composed of the following tree structure (the Centos/RHEL folder structure is shown here; Ubuntu is similar):

```
├── common
│   ├── sectools
├── <architecture>
│   ├── rpm
│   │   ├── rpm
│   │   │   ├── qaic-fw-<version>.el7.x86_64.rpm
│   │   │   ├── qaic-kmd-<version>.el7.x86_64.rpm
│   │   │   └── qaic-rt-<version>.el7.x86_64.rpm
│   │   ├── rpm-docker
│   │   │   └── qaic-rt-docker-<version>.el7.x86_64.rpm
│   │   ├── install.sh
│   │   ├── Notice.txt
│   │   └── uninstall.sh
│   ├── deb
│   │   ├── deb
│   │   │   ├── qaic-fw_<version>.deb
│   │   │   ├── qaic-kmd_<version>-devel_amd64.deb
│   │   │   └── qaic-rt_<version>_amd64.debm
│   │   ├── install.sh
│   │   ├── Notice.txt
│   │   └── uninstall.sh
│   ├── test_suite
```

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 01:50:29 GMT mixd2@lenovo.com

```
│   │   ├── pcietool
└── └── └── powerstress
```

Where <architecture> is x86_x64 or aarch64.

- Run the install.sh script as root or with sudo to install with root permissions.
  - cd <architecture>/<centos/ubuntu>
  - For Hybrid boot cards: sudo ./install.sh --auto_ubgrade_sbl
  - For Flashless boot cards: sudo ./install.sh

    NOTE Installation may take up to 15 minutes.
- On successful installation of the platform SDK, the contents are stored to the /opt/qti-aic path:

  ```
  dev  drivers  examples  exec  firmware  services  test-data  tools
  ```

- Verify card health:
  - Check card health using the qaic-util utility. The following command should output "Ready" for all cards present in the system:

    ```
    sudo /opt/qti-aic/tools/qaic-util -q | grep Status
    ```

  - Check other parameters such as Max Link Speed, Device capabilities, etc. by running the qaic-util tool per card in the system.

    ```
    sudo /opt/qti-aic/tools/qaic-util -q
    ```

2.2.5 Apps SDK Installation

The Apps SDK can be installed only on x86 hosts. Two modes are supported – Standalone (simulation on CPU) and Overlay (AIC hardware). For Overlay mode, the Platform SDK is also required.

Qualcomm Confidential - May Contain Trade Secrets 2023-08-03 04:50:29 GMT mixd2@lenovo.com

For Arm-based Qualcomm platforms, models are first compiled on x86 with the Apps SDK. The compiled binary is transferred to the Arm host for loading and execution by the Platform SDK on AIC hardware.

- Log in to https://www.qualcomm.com/products/technology/processors/cloud-artificial-intelligence/cloud-ai-100#Overview. First time users need to register.

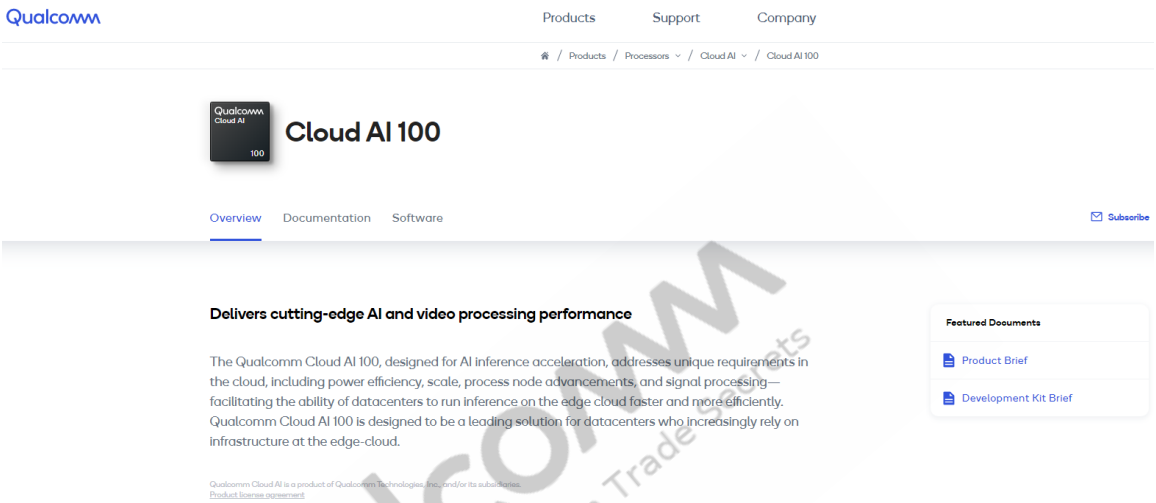

Qualcomm
Products
Support
Company
Products / Processors / Cloud AI / Cloud AI 100
Cloud AI 100
Overview
Documentation
Software
Subscribe
Delivers cutting-edge AI and video processing performance
The Qualcomm Cloud AI 100, designed for AI inference acceleration, addresses unique requirements in the cloud, including power efficiency, scale, process node advancements, and signal processing—facilitating the ability of datacenters to run inference on the edge cloud faster and more efficiently. Qualcomm Cloud AI 100 is designed to be a leading solution for datacenters who increasingly rely on infrastructure at the edge-cloud.
Featured Documents
Product Brief
Development Kit Brief

Figure 2-3 Cloud AI 100 website

- Click on the Software tab.
- Under Software tools, click on Cloud AI (AIC) Apps Software Development Kit (SDK) and download the SDK.

Notes:

1. 'Software packages' may provide additional versions of the Apps SDK that can be downloaded and installed.
2. You must be able to log in as root or use `sudo` to have root permissions to complete installation.
3. For installation of a new version of the Apps SDK, the previous version must be uninstalled first using the uninstall.sh script executed as root or sudo: `sudo ./uninstall.sh`

- Install the downloaded Apps SDK:
  - Unzip the download zip file to a work directory
  - `cd` to the work directory

  The Apps SDK is composed of the following files:

```
<Apps-SDK>
├── dev
│   ├── hexagon_tools
│   ├── inc
│   └── lib
│       └── x86_64
│           └── apps
├── examples
│   ├── apps
```

```
│   └── scripts
├── exec
│   └── qaic-exec
├── tools
│   ├── docker-build
│   ├── package-generator
│   ├── qaic-pytools
│   ├── qaic-version-util
│   ├── rcnn-exporter
│   └── smart-nms
├── install.sh
├── integrations
│   ├── qaic_onnxrt
│   └── triton
├── Notice.txt
├── scripts
│   └── qaic-model-configurator
├── versions
│   └── apps.xml
└── uninstall.sh
```

- Run the install.sh script as root or with sudo to install with root permissions.

  ```
  sudo ./install.sh
  ```

- On successful installation of the Apps SDK, the contents are stored to the /opt/qti-aic path under the dev and exec directories:

  ```
  dev exec integrations scripts
  ```

- Check the Apps SDK version with the following command:

  ```
  cat ./versions/apps.xml
  ```

- Apply chmod commands

  ```
  sudo chmod a+x /opt/qti-aic/dev/hexagon_tools/bin/*
  sudo chmod a+x /opt/qti-aic/exec/*
  ```

Setting the Environment

For using the executables provided in the Apps SDK, some supporting libraries are necessary. These are searched in the system in the following order of priority:

1. $AIC_COMPILER_LIB_DIR and $AIC_COMPILER_TOOLS_DIR environment variables.
2. `/opt/qti-aic/dev/lib/x86_64/apps/`,`/opt/qti-aic/dev/lib/x86_64/`, and`/opt/qti-aic/dev/hexagon_tools/`
3. The current, hardcoded compiler locations.

The above environment variables should be set on your system before using the compiler tools as follows:

```
$ export AIC_COMPILER_LIB_DIR=/opt/qti-aic/dev/lib/x86_64/apps:/opt/qti-aic/dev/lib/x86_64/
$ export AIC_COMPILER_TOOLS_DIR=/opt/qti-aic/dev/hexagon_tools/
```

2.3 Docker and Kubernetes support

Machine learning applications built for the Cloud AI 100 Accelerator can be containerized with Docker and deployed with Kubernetes. The following figure shows a sample Kubernetes deployment.

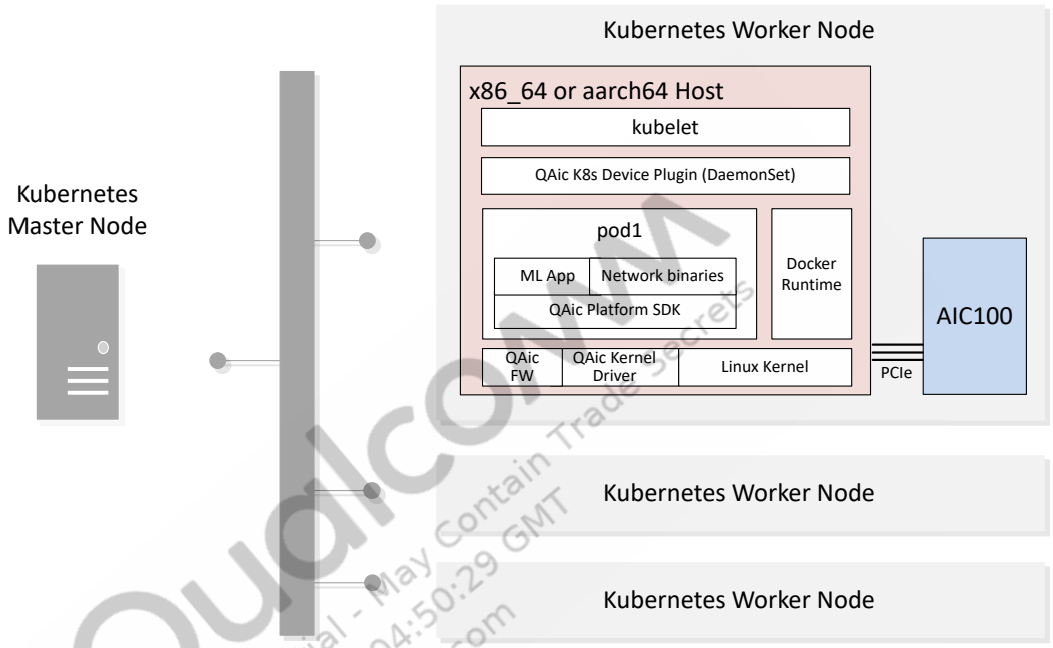


Figure 2-4 Kubernetes cluster

The following sections cover:

- Creating a Docker container with access to Cloud AI 100 Accelerators card
- Creating a deployment with Kubernetes

2.3.1 Qualcomm deliverables

Qualcomm provides the following Docker and Kubernetes scripts for Qualcomm Cloud AI 100:

- QAic docker build script

  Qaic-docker build scripts are released as part of the Apps SDK release.

- QAic Kubernetes device plugin

  The QAic Kubernetes device plugin “qaic-k8s-device-plugin-1.0.tar.gz” is provided via me2u download.

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:50:29 GMT qixd2@lenovo.com

Create a support case at https://support.qualcomm.com with the following problem areas to receive the files:

- Problem Area 1: BSP/HLOS
- Problem Area 2: AI Accelerator
- Problem Area 3: Installation

2.3.2 QAic docker

The Docker scripts are located in the Apps SDK in the `tools/docker-build` folder. The scripts to build a QAic docker image are composed of the following structure:

```
├── build_image.sh
├── caffe
│   ├── detection-output.patch
│   ├── Dockerfile.ubuntu18
│   ├── makefile.config.patch
│   └── mkldnn.patch
├── config
│   ├── aarch64
│   │   ├── Dockerfile.centos8
│   │   └── Dockerfile.ubuntu18
│   ├── qaic_pytools
│   │   ├── aimet.dockerfile
│   │   └── pytools.dockerfile
│   ├── qms_agent
│   │   └── qms_agent.dockerfile
│   └── x86_64
│       ├── Dockerfile.centos8
│       ├── Dockerfile.triton
│       └── Dockerfile.ubuntu18
├── README.md
```

Qualcomm Confidential May Contain Trade Secrets 2023-03-13 04:50:29 GMT mlx12@lenovo.com

NOTE The ubuntu18 config file works for ubuntu20 as well.

Build a QAic docker image

1. Download Apps SDK and Platform SDK from the Qualcomm site.
2. Unzip Apps SDK.

   The build scripts are under `/tools/docker-build/`.

   Example:

   ```
   unzip qaic-apps-1.8.0.124.zip
   cd /qaic-apps-1.8.0.124/tools/docker-build/
   ```

The QAic docker build script creates a new docker image with Cloud AI 100 Apps SDK and Cloud AI 100 Platform SDK with a specific OS. (Run `sudo su` before running this with bash.)

The following command builds the docker image:

```
./build_image.sh --apps-sdk <path to apps-sdk>/qaic-apps-<version>.zip --
platform-sdk <path to platform-sdk>/qaic-platform-sdk-<version>.zip --tag
"<tag name>" -os <ubuntu18> --install-qaic-pytools
```

NOTE The `ubuntu18` docker config file also works for ubuntu20.

Example:

```
./build_image.sh --apps-sdk ~/qaic-apps-1.8.0.124.zip --platform-sdk ~/qaic-
platform-sdk-1.8.0.124.zip --tag "1.8.0.124" --os ubuntu18 --install-qaic-
pytools
```

To check the docker image created with the above script:

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
qran-pytools-ubuntu18-x86_64	1.8.0.124_JN	0ca29bc4842a	28 hours ago	11.4GB
qran-ubuntu18-x86_64	1.8.0.124_JN	73f5f69b8f74	28 hours ago	8.18GB

Create a container

The following command runs the container using the docker image created above:

```
$ docker run -dit --name <container name> --mount type=bind,source=<folder on
host>,target=/<folder name in container> --publish <remote machine
port>:<docker container port>--device=<AIC device that want to pass to the
container> <docker image name>:<tag>
```

Example:

```
$ sudo docker run -dit --name JN_SAMPLES --mount type=bind,source=/
home/,target=/workspace/ --publish 8888:8888 --device=/dev/qaic_aic100_1 qran-
pytools-ubuntu18-x86_64:1.8.0.124_JN
```

Check the container that got created:

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ef37ada92065	qran-pytools-ubuntu18-x86_64:1.8.0.124_JN	"bash"	21 hours ago	Up 21 hours	0.0.0.0:8888->8888/tcp, :::8888->8888/tcp	dr_18124_notebooks

Execute the container

Execute the container created in the above step using its container ID:

- `docker exec -it ef37ada92065 /bin/bash`

Or

- `docker exec -it JN_SAMPLES /bin/bash`

This command launches a bash shell within the container.

Query the AIC100 devices in the container using qaic-util tool:

```
/opt/qti-aic/tools/qaic-util -q
```

Run a quick inference test from the shell to validate the setup:

```
/opt/qti-aic/exec/qaic-runner -d 0 -t /opt/qti-aic/test-data/aic100/v2/1nsp/
1nsp-quant-resnet50/
InferenceCnt 40 TotalDuration 63808us BatchSize 1 Inf/Sec 626.881
```

NOTE Update the `-d` option in the above command based on the device passed to the container.

Build docker for aarch64 from x86_64

To build an aarch64 image from an x86_64 machine, you need to set up binary translations. Use the following commands to install:

- Install support packages
  - Ubuntu:

    ```
    sudo apt-get install qemu binfmt-support qemu-user-static
    ```

  - CentOS:

    ```
    sudo yum install qemu binfmt-support qemu-user-static
    ```

- Set up docker runtime

  ```
  docker run --rm --privileged multiarch/qemu-user-static --reset -p yes
  ```

2.3.3 Kubernetes

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services. A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node. The worker node(s) host the pods that are the components of the application workload. The master node manages the worker nodes and the pods in the cluster.

QAic K8s device plugin is provided to allow a Kubernetes master to manage AIC100 devices in the cluster.

QAic K8s device plugin

The QAic device plugin for Kubernetes is a daemonset which allows you to automatically:

- Report the number of AIC100 devices on each node in your cluster.
- Monitor the health status AIC100 devices.
- Be capable to run containers with AIC100 devices.
- Provides a way to share device across containers.

The command to build the device plugin image is:

```
$ bash build_image.sh -a <architecture>
```

Example:

```
$ bash build_image.sh -a aarch64
```

To know architecture, run 'uname -m', the above command supports “aarch64” and “x86_64”.

NOTE A cross architecture build requires docker setup for binary translation.

To install device plugin on master node:

```
$ kubectl create -f <device plugin file name>
```

Example:

```
$ kubectl create -f qaic-device-plugin.yml
daemonset.apps/qaic-device-plugin-daemonset created
```

To view available qaic resources:

```
$ kubectl describe node | grep qaic
```

Argument details

Table 2-2 QAic K8s device plugin build script argument details

Argument	Description
-a	<Architecture type(x86_64/aarch64)>
--help	For help

2.4 Deployment examples

2.4.1 Deploy with mini-kube

Prerequisites

Docker and mini-kube installed on host machine with AIC100 card.

Start mini-kube

Make sure mini-kube is up and running,

- As a root, start mini-kube using the following command (this will use the docker registry of your local machine):

  ```
  $ minikube start --driver=none
  ```

- Check that '`/var/lib/kubelet/device-plugins/kubelet.sock`' exists.
- Run the following command to check that the node shows up:

  ```
  $ kubectl describe node
  ```

QAic K8s device plugin

Build the device plugin docker image and install using device plugin build script, qaic-device-plugin.yml file.

- Get QAic k8s device plugin build script and .yml files.
- Use the following command to build the device plugin image:

  ```
  $ bash build_image.sh -a  aarch64
  ```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

- Check the docker image created above:

```
$ docker images
REPOSITORY                TAG       IMAGE ID         CREATED         SIZE
qaic-k8s-device-plugin    v1.0.0    f8de6163fc79     5 days ago      82.2MB
```

- To install the device plugin, use the following command:

```
$ kubectl create -f qaic-device-plugin.yml
daemonset.apps/qaic-device-plugin-daemonset created
```

- To view available qaic resources:

```
$ kubectl describe node | grep qaic

qualcomm.com/qaic:  4
qualcomm.com/qaic:  4
  kube-system                 qaic-device-plugin-daemonset-d2xnj     0
(0%)        0 (0%)      0 (0%)           0 (0%)         160m
  qualcomm.com/qaic  1           1
```

Running deployment

Create and run the deployment and verify by querying the device and running ResNet-50 workload:

- To create the deployment:

```
$ kubectl apply --filename=./deploy-qranium-single.yaml --record=true
deployment.apps/qranium-deployment created
```

- Check if the deployment is running:

```
$ kubectl get deployments
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
qranium-deployment   1/1     1            1           160m
```

- Run the following command to get the running containers:

```
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
qranium-deployment-69c588c54f-vrsl6   1/1     Running   0          7s
```

- Run the following to get a shell in the pod:

```
$ kubectl exec --stdin --tty qaic-device-plugin-daemonset-vrsl6 -- /bin/
bash
bash-4.4$
```

- From the container, query device to display card details:

```
bash-4.4$ /opt/qti-aic/tools/qaic-util -q

LRT QC_IMAGE_VERSION: LRT.AIC.1.4.1.1.12
LRT IMAGE_VARIANT: LRT.AIC.REL
Number of devices: 1
QID 0
        Status:Ready
        PCI Address:0002:01:00.0
```

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:59:23 GMT hlxd2@lenovo.com

```
PCI Info:Unassigned class [ff00] Qualcomm Device a100
HW Version:0.1.0.0
HW Serial:0x9ae51a65
FW Version:1.1.12
FW QC_IMAGE_VERSION:QSM.AIC.1.1.12
FW OEM_IMAGE_VERSION:
FW IMAGE_VARIANT:AIC100.REL
NSP Version:1.1.5
NSP QC_IMAGE_VERSION:NSP.AIC.1.1.5
NSP OEM_IMAGE_VERSION:
NSP IMAGE_VARIANT:aic100.nsp.prodQ
Compiler Version:0
Dram Total:8122 MB
Dram Free:8122 MB
Dram Fragmentation:0.00%
Vc Total:16
Vc Free:16
Nsp Total:8
Nsp Free:8
Peak Dram Bw:0.0
Peak Sram Bw:0.0
Peak PcieBw:0.0
MCID Total:3072
MCID Free:3072
Semaphore Total:32
Semaphore Free:32
Constants Loaded:0
Constants In-Use:0
Networks Loaded:0
Networks Active:0
```

- To run ResNet-50 workload, test data for v1 or v2 hardware versions are located at:

  `/opt/qti-aic/test-data/aic100/<v1 or v2>`

- Determine hardware version using qaic-util:

  ```
  bash-4.4$ /opt/qti-aic/tools/qaic-util -q | grep "HW Version"
  HW Version:0.1.0.0
  ```

- From the container run the quantized ResNet-50 workload using qaic-runner:

  ```
  bash-4.4$ /opt/qti-aic/exec/qaic-runner -t /opt/qti-aic/test-data/
  aic100/v1/1nsp/1nsp-quant-resnet50/ -n 5000 -a 8

  Device: ID 0
  Test path: /opt/qti-aic/test-data/aic100/v1/1nsp/1nsp-quant-resnet50/
  Number of activations: 8
  InferenceCnt 4960 TotalDuration 3227623us Inf/Sec 1536.73
  ```

Qualcomm Confidential - May Contain Trade Secrets 2023-05-03 04:50:29 GMT mixd3@lenovo.com

3 Release Notes

Qualcomm Cloud AI 100 Software Release Notes (80-PT790-1)

Qualcomm
Confidential - May Contain Trade Secrets
2023-03-03 04:50:29 GMT
mixd2@lenovo.com

4 Neural Network Support

AI 100 supports over 150+ networks across multiple categories – image classification, object detection, semantic segmentation, pose estimation, face detection, natural language processing, GAN, networking, recommender systems, and ADAS. ADAS networks supported can be categorized into perception with and without LiDAR, pedestrian/lane detect, traffic light detect, semantic segmentation, object detection, trajectory prediction, point cloud, etc. The list of networks onboarded on AI 100 is growing rapidly. This is not an exhaustive list but a representation of the breadth of neural networks supported on AI 100.

Computer Vision			
Image classification	**Object detection**	**Semantic segmentation**	**Others**
3DCNN	CenterNet	DeepLab-V3	**Face Detection**
AlexNet	Detectron2-COCO	DeepLab-V3+	CenterFace
DenseNet-121	DLA-CenterNet34	MaskRCNN	InsightFace
DenseNet-169	EfficientDet-D0/D7	PolarSeg	MobileFaceNet
EfficientNet-B0/B7	FasterRCNN-R50-C4-VOC	ShapeMask	RetinaFace
GoogleNet	FasterRCNN-R50-FPN-VOC	SqueezeSegV3	SCRFD
Inception-18	LPRNET	U-Net	**Point Cloud**
Inception-V2	MaskRCNN		PointNet
Inception-V3	ResNet18-CenterNet		Second Network
Inception-V4	ResNet18-SSD		**Pose Estimation**
MobileNet-V1	ResNet34-SSD		HRNet
MobileNet-V2	RetinaNet		
MobileNet-V3	SSDLite-MobileNetV2		
RegNet-FAIR	SSD-MobileNetV1		
ResNet-101	SSD-MobileNetV2		
ResNet-18	YoloV2		
ResNet-32	YoloV3		
ResNet-34	YoloV3-Tiny		
ResNet50-V1.5	YoloV4		
ResNet50-V2	YoloV5		
ResNext-101	YoloX		
ResNext-50			
ShuffleNet-V1			
ShuffleNet-V2			
SqueezeNet			

Qualcomm Confidential - May Contain Trade Secrets 2025-03-03 04:50:29 GMT nkv2@lenovo.com

Computer Vision			
Image classification	**Object detection**	**Semantic segmentation**	**Others**
SqueezeNet-V1.1 VGG-16 VGG-19 ViT WideResNet-50			

Natural language processing	ADAS	Others
Alberta	Perception with LiDAR	**Recommended systems**
BART	Perception without LiDAR	DLRM 10M
Bert Base	Traffic sign recognition	**GANs**
Bert Dense Pruned	Sensor blockage detection	SRGAN
Bert Large	Transformer (Camera , LiDAR)	**IP Networking**
Bert QAT	LiDAR segmentation	IUPG
Bert3	Image classification	Classifier
BigBird	Object detection	
CamemBERT	Sequence classification	
CharRNN	Semantic segmentation	
CLIP	Point cloud	
CodeBERT	Trajectory prediction	
CodeBERTa	Vehicle pose/orientation estimation	
CodeGen		
Deberta		
DebertaV3		
Deepspeed MOE		
DistilBART		
DistilGPT2		
DistillBert		
DistillRoberta		
FastTransformer		
Glide Text2image		
GPT2		
GPT2-Large		
GPT2-Medium		
GPT2-Neo		
InCoder		
Indic-BERT		
IndicNER		
LongFormer Classification		
MiniLM		
MPNet		

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:56:29 GMT mixd2@lenovo.com

Natural language processing	ADAS	Others
OPT		
PhoBERT		
Roberta-Base		
Roberta-Large		
Turing3		

Qualcomm Confidential - May Contain Trade Secrets 2023-03-03 04:50:29 GMT mixd2@lenovo.com

5 Quick Start Guide

This section provides AI 100 users (system integrators, ML application developers, etc.) several examples to get started with the AI 100 card and inference toolchain. This includes showcasing of end-to-end inference workflows, card setup, individual tools/features, etc.

Several more examples are in the `/opt/qti-aic/examples` folder. This requires the Platform and Apps SDKs to be installed on the server. Examples include:

- Source code for the inference tools in cpp
- Sample inference code in `cpp/python`
- Sample cpp code for key inference features, such as over-subscription and multi-model execution

5.1 Performance tuning overview

The following list summarizes key compile and runtime parameters that affect the performance of throughput and latency (Qualcomm Cloud AI 100 Deep Learning Inference Workflow Overview discusses several other parameters):

- Core

  Compile flag. This parameter refers to the Neural Signal Processor (NSP) in the AI 100 SoC. The number of NSPs available depends on the card SKU. The user defines the number of cores used for a neural network.

- Activation/instance

  Runtime flag. This parameter refers to an instance of the compiled neural network binary running on a set of cores. Multiple activations can execute concurrently on AI 100 (on distinct set of cores). The total number of activations depend on the total number of cores available and cores used per activation. The user can choose to run multiple activations and yet have cores left over for execution on other networks

- Batch Size

  Compile flag. This parameter refers to the number of input elements that are inferred concurrently by an activation. The user also defines this parameter.

- MOS – Maximum output splitting

  Compile flag. This parameter refers to the number of cores across which an output channel (and associated weights) is split. The default value provides good out-of-box performance in most cases.

Qualcomm Confidential – May Contain Trade Secrets 2024-03-03 04:17:29 GMT mixz@lenovo.com

- OLS – Overlap split

  Compile flag. This parameter enables output splitting to improve core level parallelism; e.g., vector and tensor units. The default value provides good out-of-box performance in most cases.

- Set-size

  Runtime flag. This parameter denotes the number of inferences that can be queued per activation. This helps hide host side overhead by pipelining inferences.

The model configurator tool is the best way to identify the set of parameters for best latency or throughput for a network. Consider the following high-level trade-offs while tuning for performance:

- Cores and synchronization overhead

  For a given activation with fixed batch size, increasing the number of cores from 1 will typically improve performance. Increasing the number of cores beyond the optimum number causes performance to drop due to increased synchronization overhead.

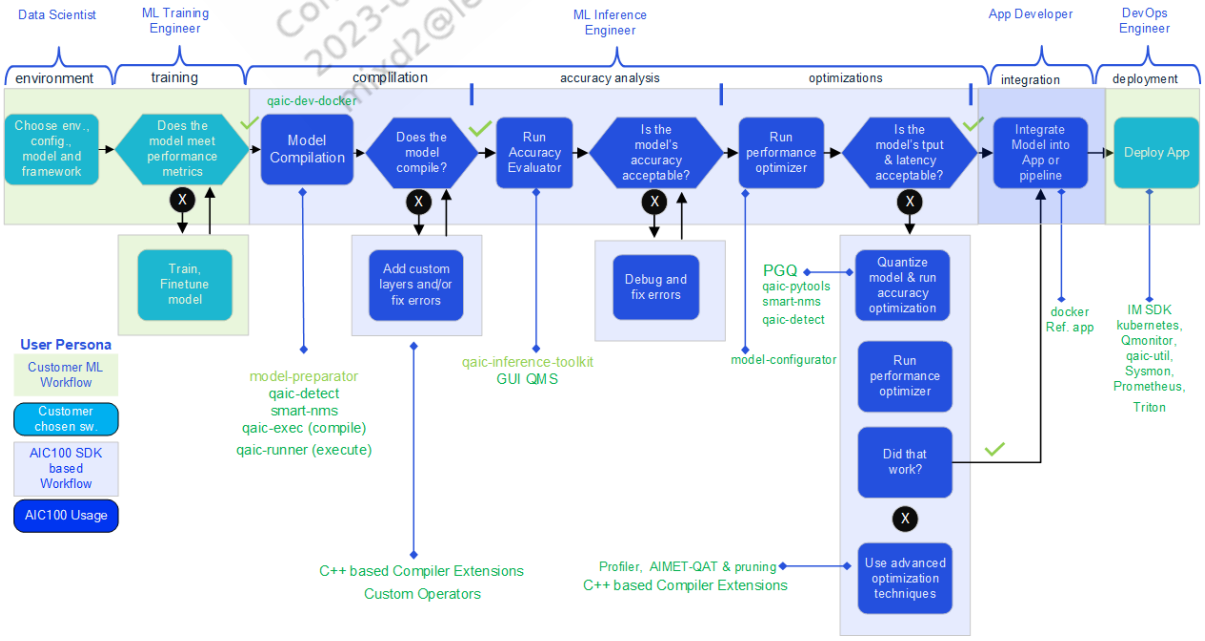
- Batch size and on-chip memory

  For an activation of a fixed number of cores, increasing the batch size from 1 will typically improve throughput (inferences per second). Increasing beyond the optimum BS will cause the throughput to drop. Performance on AI 100 is maximized when the DDR BW is minimal (data is resident in on-chip memory). Increasing BS beyond the optimal setting increases the DDR BW (spillage) leading to lower performance.

5.2 Inference workflow overview

The following diagram illustrates the end-to-end inference workflow on AI 100. It shows a single pass/iteration, but the workflow can be iterative across compile, accuracy analysis, and optimization steps.

This diagram lists the tools, such as qaic-exec, that are used in different aspects of the inference workflow.



MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.3 Jupyter notebooks

The following quick start guides are available to download outside of the SDKs.

- Card setup guide
- Model onboarding and tool usage examples
- End-to-end inference workflow illustrations
  - DenseNet
  - Yolov5

Card setup guide

Target audience: This is the first step for users who want to evaluate DNN performance on an x86/ARM platform using Cloud AI 100 cards.

Takeaways: Users will learn to set up device configurations, debug fundamental issues, determine card health, and use the capabilities of Cloud AI 100 platforms.

HD-10004-AI00

Model onboarding and tool usage examples

Seehttps://www.qualcomm.com/products/technology/processors/cloud-artificial-intelligence/cloud-ai-100?classification=Quick+Start+Guide#Documentation

End-to-end inference workflow illustrations

- DenseNet HD-10004-AI20
- Yolov5 HD-10004-AI21

Qualcomm Confidential – May Contain Trade Secrets 2023-03-06 04:50:09 GMT mixd2@lenovo.com